Installation and Reference Guide
High Availability Virtualization
using HA-Lizard
with
Citrix XenServer and
Xen Cloud Platform (XCP)

Version 1.6.41

The information in this document and any product or service specifications referred to herein are subject to change without notice.

XenServer, XenCenter, Xen Cloud Platform and XCP are registered trademarks or trademarks of Citrix System, Inc and Xen.org
ILO and Integrated Lights Out are registered trademarks of the Hewlett Packard Corporation.

No part of this document may be reproduced, copied, altered or transmitted in any form or by any means, electronic, mechanical or otherwise for any purpose whatsoever, without the express written permission of the Copyright owner.

The information provided in this document is intended as a guide only and is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

No support is provided as part of the information in this document or any related software. Contact the project sponsor, Pulse Supply (www.pulsesupply.com), for details on support offerings.

Document Copyright © 2013 Pulse Supply
All rights reserved.

IMPORTANT

##
!!! HA-Lizard is free software: you can redistribute it and/or modify
!!! it under the terms of the GNU General Public License as published by
!!! the Free Software Foundation, either version 3 of the License, or
!!! (at your option) any later version.
!!!
!!! HA-Lizard is distributed in the hope that it will be useful,
!!! but WITHOUT ANY WARRANTY; without even the implied warranty of
!!! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
!!! GNU General Public License for more details.
!!!
!!! You should have received a copy of the GNU General Public License
!!! along with HA-Lizard. If not, see <http://www.gnu.org/licenses/>.
##

Table of Contents

1. HA-Lizard Version 1.6.41 6
- Purpose 6

2. Installation 7
- Installing 7
- Removing 7
- Installing on Debian or other Dom0 based systems 7

3. Configuring 9
- ha-cfg tool 9

insert 9

remove 9

log 9

get 10

set 10

get-vm-ha 10

set-vm-ha 10

Status 10
- Global Configuration 11
- Override Configuration 11
- Configuration Parameters 12

HA Monitor Configuration 12

MONITOR_MAX_STARTS 12

MONITOR_KILLALL 13

MONITOR_DELAY 13

MONITOR_SCANRATE 13
XC_FIELD_NAME 13
OP_MODE 13
GLOBAL_VM_HA 14
ENABLE_LOGGING 14
DISABLED_VAPPS 14
XE_TIMEOUT 14
XAPI_COUNT 14
XAPI_DELAY 14
SLAVE_HA 14
PROMOTE_SLAVE 15
SLAVE_VM_STAT 15
Email Alert Settings 15
Fencing Configuration Settings 15
Additional Local Configuration Parameters 17
4. Modes of Operation 19
5. Managing HA Services 20
- HA-Lizard System Service 20

HA Watchdog Service 20
Disabling the Watchdog Service 20
- HA Monitor 20
- Managing HA for the Pool 21

Cluster Management and Fencing 22
- Pool Cluster Management 22

Cluster Failover and Recovery Logic 22
- Split Brain Handling 22
- Fencing and STONITH 23

ILO Fencing 23

Pool Fencing 24

XVM Fencing 24
- Adding your own Custom Fencing Script 26

6. Managing via XenCenter 27
7. Application and Settings Examples 29
- Auto-Start VMs – Without Host High Availability 29
- Auto-Start VMs – With Host High Availability 30
- 2-Node Pool – With Host High Availability 31

8. Miscellaneous 31
- Dependencies and Compatibility 31
- Security and Ports 32
- Things to Know 32
- Future Improvements 33
- Support 33

1. HA-Lizard Version 1.6.41

- **Purpose**

HA-lizard provides complete automation for managing Xen server pools which utilize the XAPI management interface and toolstack (as in Xen Cloud Platform and XenServer). The software suite provides complete HA features within a given pool. The overall design is intended to be lightweight with no compromise of system stability. Traditional cluster management suites are not required. HA is provided with built in logic for detecting and recovering failed services and hosts.

HA features provided:

- Auto-start of any failed VMs
- Auto-start of any VMs on host boot
- Detection of failed hosts and automated recovery of any affected VMs
- Detect and clean up orphaned resources after a failed host is removed
- Removal of any failed hosts from pool with takeover of services
- Fencing support for HP ILO, XVM and POOL fencing (forceful removal of host from pool)
- Split brain prevention using heuristics from external network points and quorum
- HA support for pools with two hosts
- Structured interface for simple “bolt-on” of fencing scripts
- Dual operating modes for applying HA to appliances or individual VMs
- Ability to exclude selected appliances and VMs from HA logic
- Auto detection of host status allows for safely working on hosts without disabling HA
- Centralized configuration for entire pool stored in XAPI database
- Command-line tool for managing global configuration parameters
- Parameter override available per host for custom configurations
- HA can be enabled/disabled via command line tool or graphical interface (like XenCenter)
- Extensive Logging capabilities to system log file
- Email alerting on configurable triggers
- Dynamic cluster management logic auto-selects roles and determines recovery policy
- No changes to existing pool configuration required. All logic is external.
- **No dependencies** – does not compromise pool stability or introduce complex SW packages. Designed to work with the resident packages on a standard XCP/XenServer host

Development is well tested and based on Xen Cloud Platform (XCP) version 1.6 and XenServer 6.1

2. Installation

An included installer is designed for systems based on Centos DomOs. Manual installation is required for Debian based DomOs. The following installation instructions will work with a standard ISO install of XCP version 1.6 or XenServer version 6.1.

- **Installing**

- Copy the source tarball into a temporary location (ex. /tmp/)
- Extract its contents and move into the extracted folder
  tar –zxvf ha-lizard-1.6.41*.tgz
- Move into the “scripts” folder
  cd ha-lizard-1.6.41*/scripts
- Run the installer
  ./install

The installer will check if sendmail and mailx packages are installed on the server. These are only required for email alerts. Skip the installation of these packages if email alerting is not required.

The installer will install the default pool parameter set in the XAPI database. This step is only required on a single host.

Once the installer is completed, HA and watchdog services will be started. Although these services are running, HA is disabled for the pool by default. HA can then be enabled via the command line tool <ha-cfg> to enable HA once installation has been completed on all hosts within a pool.

- **Removing**

An uninstall script is provided for Centos based systems. This will completely remove all components from the host. If uninstalling on a single host, skip the step which removes the settings stored in the XAPI database.

- **Installing on Debian or other Dom0 based systems**

A non-CentOS based Dom0 is not supported. The following steps will get most of the functionality working with the exception of init and watchdog scripts which are critical in HA recovery operations. During a recovery operation it is very likely that the HA service will be intentionally killed due to slaves being in emergency mode upon loss of communication with a master. A watchdog is expected to restart HA within a few seconds, only this time hosts use data stored in their local cache to recover a pool rather that attempt communication with XAPI which may be unresponsive.

- Copy the source tarball into a temporary location (ex. /tmp/)
- Extract its contents
  tar –zxvf ha-lizard-1.6.41*.tgz
- Rename and copy the source directory to /etc/
  mv ha-lizard-1.6.41* ha-lizard
  cp –r ha-lizard /etc/
- Insert the default pool settings into the XAPI database (this steps needs to be completed only one time on any of the pool hosts)
  /etc/ha-lizard/scripts/ha-cfg insert
- Insert custom field used to toggle HA on/off
  xe pool-param-add uuid=`xe pool-list –minimal` param-name=other-config XenCenter.CustomFields.ha-lizard-enabled=false
- Insert custom field used to track the autopromote-uuid
  xe pool-param-add uuid==`xe pool-list –minimal` param-name=other-config autopromote_uuid=""
- Start the service in the background
  /etc/ha-lizard/scripts/ha-lizard.mon &
  IMPORTANT: To ensure proper operation on non CentOS based Dom0, a watchdog service must be added to watch ***/etc/ha-lizard/scripts/ha-lizard.mon*** and restart it upon failure.

3. Configuring

- **ha-cfg tool**

A command line tool is provided with the package for managing configuration and enabling/disabling HA for the entire pool. Usage is displayed with “ha-cfg –help”.

```
Pool Configuration and Monitoring Tool for xen/xapi High Availability
Usage: ha-cfg <action> <values for selected action>

Available actions:
<insert>:          Inserts default values into xapi pool database.
                   Use with caution. Intended for a fresh install
                   or repair of existing installation.
<remove>:          Removes all custom fields from xapi pool database
                   Use caution. Intended to repair existing installation.
<log>:             Watch HA log output in real time for
<get>:             Lists all HA params and values stored in pool database
<set>:             Sets the specified parameter to the specified value.
                   ex. [ha-cfg set MAIL_ON 0] disables email alerts
                   Do not use spaces in any configuration values.
                   Usage <ha-cfg set [parameter] [value]>
<get-vm-ha>:       Lists all pool VMs and status of HA enabled for each VM
<set-vm-ha>:       Enable/Disable HA for the specified VM
                   Only needed when configured to individually select managed VMs
                   Usage <ha-cfg set-vm-ha [vm-name-label] [true or false]>
<status>:          Displays the HA operational status of the pool
                   and enables/disables HA for the pool.
```

insert

The default Centos based installer will run this tool with the “insert” option. This action should only be completed one time for an entire pool. Insert will create all the required global parameters and write them to the XAPI database with default values. This tool is helpful when installing on non-Centos based hosts where the default installer cannot be used. Running this manually will initialize all default values. Running this over an existing installation may overwrite the global pool parameters.

remove

The “remove” option will erase all global parameters from the XAPI database. This will essentially “clean” the database and should be performed only when removing HA from a pool manually. The provided uninstall script will execute this action by default.

log

The “log” action will invoke a simple logging utility that displays system logs written by HA. It is a convenient way to watch system behavior and useful in troubleshooting.

get

The “get” action lists all of the global parameters and their values. This is useful for checking the current configuration and also for retrieving the correct syntax of a parameter to be used in a “set” command

set

The “set” action is used modify the value of any global parameter. Set requires additional command line arguments as follows:

“ha-cfg set <parameter> <value>”

Where <parameter> is the name of the parameter to be updated (correct syntax of the parameter can be retrieved by invoking “ha-cfg get”) and <value> is the new value for the given parameter. Check the configuration file documentation for a list of supported values.

get-vm-ha

The “get-vm-ha” action will list all VMs within a pool and display whether HA is enabled for each VM. Additional information is provided depending on the pool setting context. This tool is useful for quickly determining the HA status of any VM.

```
Pool OP_MODE is set to 2 - all VMs that require HA should have HA enabled.
Use [ha-cfg set-vm-ha ] to set HA status on VMs.

HA-Enabled  VM-UUID                                VM-Name
false       6c792e1a-d05a-0d54-4586-40d730912c25  cent1
true        a2bb1c52-1960-fd54-7a2f-4ba2fbd3e774  cent4
true        10e740e7-8ba6-85d0-7f57-0b727a6ec98b  cent3
unset       a89b52cd-a141-2de7-2603-56c83daec6d4  cent2
```

set-vm-ha

The “set-vm-ha” action is used to enable or disable HA for the specified VM name-label. The passed in name-label must exactly match the name of the VM. Value must be set to true or false.

```
> ./ha-cfg set-vm-ha cent4 false
UUID for cent4 found: a2bb1c52-1960-fd54-7a2f-4ba2fbd3e774
Updating VM:cent4 UUID: a2bb1c52-1960-fd54-7a2f-4ba2fbd3e774 to status: false
Success: VM: cent4 updated to HA status: false
Wait at least 45 seconds for changes to be applied
```

Important: No spaces should ever be used for any of the values.

Status

The “status” action displays whether HA is enabled or disabled for the pool and allows for the toggling of the status. This should be used to enable/disable HA for the pool as shown below.

```
################################################################################################
## Checking whether   HA   is enabled for pool: e278ae83-6801-3bde-1aee-abed73feac70 ##
## This only checks whether HA is enabled for the pool. TO check whether             ##
## the xcp-ha service is running on this host, try: service xcp-ha status            ##
################################################################################################
Pool xcp-ha-enabled Status: DISABLED
xcp-ha is disabled. Enable? <yes or Enter to quit>
yes
Success. HA enabled
```

- **Global Configuration**

Global configuration parameters are stored in the XAPI database shared by all hosts within a pool. A command-line tool, “ha-cfg”, is provided for making changes to any of the global parameters. To view the current configuration, use “ha-cfg get”. Sample output is shown below.

```
[root@xcp1 scripts]# ./ha-cfg  get
DISABLED_VAPPS=89987987:93827423984:ihrhff9:298749823
ENABLE_LOGGING=1
FENCE_ACTION=stop
FENCE_ENABLED=1
FENCE_FILE_LOC=/etc/xcp-ha/fence
FENCE_HA_ONFAIL=1
FENCE_HOST_FORGET=1
FENCE_IPADDRESS=
FENCE_METHOD=POOL
FENCE_MIN_HOSTS=3
FENCE_PASSWD=
FENCE_REBOOT_LONE_HOST=0
GLOBAL_VM_HA=1
MAIL_FROM="root@localhost"
MAIL_ON=1
MAIL_SUBJECT="SYSTEM_ALERT-FROM_HOST:$HOSTNAME"
MAIL_TO="root@localhost"
MONITOR_DELAY=45
MONITOR_KILLALL=1
MONITOR_MAX_STARTS=20
MONITOR_SCANRATE=10
OP_MODE=2
PROMOTE_SLAVE=1
SLAVE_HA=1
SLAVE_VM_STAT=0
XAPI_COUNT=5
XAPI_DELAY=15
XC_FIELD_NAME=xcp-ha-enabled
[root@xcp1 scripts]#
```

Any global configuration parameter can be updated using the “ha-cfg set” command. For example, to disable logging globally, use “ha-cfg set ENABLE_LOGGING 0”

- **Override Configuration**

By default, configuration settings for the entire pool are centrally stored and managed. A single set of configuration parameters is used across all hosts within a pool, regardless of whether a host is acting as the pool master or a slave. This makes management simple while eliminating the need to manage individual configuration files on each host within a pool.

In cases where a custom configuration is required for individual hosts within a pool, the system will allow for the override of any global settings. All override settings are stored in /etc/ha-lizard/ha-lizard.conf. By default, all settings are commented out of this file. Uncommenting a setting will override the global setting for a particular host on the next invocation of the HA script.

This is useful in cases where custom settings are required for a host that should not be shared with other hosts in the pool. One example of a good use is the SLAVE_VM_STAT setting which determines whether Slaves within the pool will try to start any VMs that are not running. By default this is disabled and only the Master is configured to start VMs that should be running. The default settings work fine for a small pool, but, in very large pools it may take some time for the Master to loop through all VMs and check their status. Enabling SLAVE_VM_STAT on a few, but not all, Slaves will speed up the process of detecting VM failures and restoring services. In this case, the override settings can be used to enable this on only selected hosts.

- **Configuration Parameters**

HA Monitor Configuration

The HA service is run by a monitoring service which runs continuously. The following Monitor settings are used to configure the behavior of the monitor. The provided installer installs and configures the Monitor with default settings that are acceptable in most cases. Once installed, the Monitor will run continuously as a service. Status can be checked with “service ha-lizard status”.

Some values may need to be changed depending on the environment. Host performance (how long it takes to iterate through HA processes launched by the Monitor) and the size of the pool should be considered when setting these parameters.

The Monitor will launch several HA processes in a loop every 45 seconds (MONITOR_DELAY). It is a good idea to watch the system logs to ensure that all the HA processes have finished running within the MONITOR_DELAY window. By increasing MONITOR_DELAY, it will take longer to detect a failure. Decreasing MONITOR_DELAY will more quickly detect failures and recover. System performance and preferences should be considered carefully.

MONITOR_MAX_STARTS

Threshold for when to assume running processes are not responding. Sets how many failed starts to wait before killing any hung processes. Use caution with this setting. It should be set relatively high as a pool recovery procedure will take considerably more time to execute causing the system to trigger logic which attempts to detect hung processes. Processes will generally never hang unless XAPI becomes totally

unresponsive. Logic is provided to abort attempts to connect to XAPI during a failure. In this case, local state files with pool state information from the last successful iteration will be used instead. Default = 40

MONITOR_KILLALL

If MAX_MONITOR_STARTS threshold is reached - set whether to kill all ha-lizard processes. Default = 1
1 = yes, 0 = no

MONITOR_DELAY

Delay in seconds between re-spawning ha-lizard. This should be adjusted to the environment. Large pools require more time for each run of ha-lizard. Default = 45

MONITOR_SCANRATE

ha-lizard will not re-spawn unless all current processes are completed. If there are active processes while attempting to start a new iteration, ha-lizard will wait the number of seconds set here before retrying. Each successive fail will increment a counter (MONITOR_MAX_STARTS) that may trigger KILLALL. Default = 10

XC_FIELD_NAME

Field name used to enable / disable HA for pool. This can also be set within XenCenter management console to enable/disable HA within the pool. To make this field visible within XenCenter, create a custom pool level field with the name set here. The configuration contents will then be visible and alterable within XenCenter. See section on configuring XenCenter for more details.

OP_MODE

Set the operating mode for ha-lizard. 1 = manage appliances, 2 = manage virtual machines

Mode 1 uses logic to manage appliances within the pool. By default, all appliances within a pool are managed by HA without requiring any additional settings. This is useful for small pools where most HA management functions can be handled from within XenCenter. Simply add VMs to appliances and those VMs are automatically managed. Any VM that is not part of an appliance is not managed by HA. This technique greatly simplifies HA management as each individual VM does not require any special settings. If some appliances should not be managed by HA, simply add the UUID of the appliances to be skipped in the DISABLED_VAPPS setting.

Managing appliances also provides a convenient way to configure startup order and delays when recovering VMs. This can be done as part of the standard appliance configuration settings.

Mode 2 provides greater flexibility by providing a mechanism for more granular control over which pool VMs are managed. Appliances are not managed in this mode. By default (when GLOBAL_VM_HA is set to 1), all VMs are automatically managed by HA. If GLOBAL_VM_HA is set to 0, then each VM within the pool must have HA explicitly enabled or disabled. This can be set from the pool GUI with access to custom configuration parameters or via the command line tool ha-cfg.

GLOBAL_VM_HA

Set whether to individually enable HA on each VM within the pool (when OP_MODE = 2)
0 = You must individually set ha-lizard-enabled to true/false for each VM within the pool
1 = ALL VMs have HA enabled regardless of setting in GUI/CLI

Default value = 1

ENABLE_LOGGING

Enable Logging 1=yes, 0=no. Logs are written to /var/log/messages. All log messages are labeled with "ha-lizard" for easy filtering. View/Filter real time logging with: “tail -f /var/log/messages | grep ha-lizard”

DISABLED_VAPPS

Specify UUID(s) of vapps that do not get automatically started by ha-lizard when OP_MODE=1 (manage appliances) Array is ":" delimited like this (UUID1:UUID2:UUID3) Leave blank if ALL vapps are managed by ha-lizard "DISABLED_VAPPS=()" Only applied when OP_MODE=1

XE_TIMEOUT

Set the maximum wait time for calls to the xe toolstack to respond. If xe does not respond within XE_TIMEOUT seconds, the xe PID will be killed. This is done to prevent xe calls from hanging in the event of a Master host failure. In the event of a Master failure, xe may hang on requests from pool slaves. This timeout will ensure that fencing the of master host is not prevented. Default setting is 5 seconds which is ample time for a well running pool. Poor performing hosts may need to set this value higher to prevent unintended process terminations during normal operation.

XAPI_COUNT

If a pool member cannot reach a pool peer, XAPI_COUNT is the number of retry attempts when a host failure is detected. If the unresponsive host is recovered before the XAPI_COUNT is reached, attempts to fence and remove the failed host will be ignored.

Default XAPI_COUNT = 5

XAPI_DELAY

If a pool member cannot reach a pool peer, XAPI_DELAY is the number of seconds to wait in between XAPI_COUNT attempts to contact the unresponsive host.

DEFAULT XAPI_DELAY = 15 seconds

SLAVE_HA

If the Pool Master cannot be reached and all attempts to reach it have been exhausted, set whether the autoselected slave will try to start appliances and/or VMs. (PROMOTE_SLAVE must also be set to 1 for this to work)

PROMOTE_SLAVE

If the pool master cannot be reached - set whether slave should be promoted to pool master (this only affects a single slave: the "autoselect" winner chosen by the former master to recover the pool). More on this topic is available in the Cluster Managemet section.

SLAVE_VM_STAT

By default, only the pool master will check the status of all VMs managed by this script and attempt to start a VM that is not in the running state. Setting SLAVE_VM_STAT to 1 will cause any pool slaves to also check all VM statuses and attempt to start any VM not in the running state. Default = 0 In a large pool many hosts may attempt to start the same VM the first host to attempt will succeed, others will be safely declined. Enabling may create many unnecessary duplicate processes in the pool.

Email Alert Settings

- MAIL_ON: Enable/Disable email alerts. 1 = enabled 0 = disabled
- MAIL_SUBJECT: Subject Line of email alert
- MAIL_FROM: The FROM email address used on email alerts
- MAIL_TO: the email address to send alerts to

Important: Do not use spaces when setting via the CLI tool. If spaces are required in the subject line, use the override configuration file instead and wrap the string in double quotes like this. “Mail Subject”

Fencing Configuration Settings

Currently Supported FENCE_METHOD = ILO, XVM, POOL

ILO is the HP Integrated Light Out management interface

XVM is intended for test environments with nested xen instances where pool domos are domus within a single xen host.

POOL does not fence failed hosts. It simply allows the forceful removal of a failed host from a pool.

The name of any custom fencing agent can also be named here. See the fencing section for details on using a custom fencing script.

- ***FENCE_ENABLED***

Enable/Disable Fencing 1=enabled 0=disabled

- ***FENCE_FILE_LOC***

Location to store and look for fencing scripts

- ***FENCE_HA_ONFAIL***

Select whether to attempt starting of failed host's VMs on another host if fencing fails 1=enabled 0=disabled

- ***FENCE_METHOD***

ILO, XVM and POOL supported. "FENCE_METHOD=ILO" - custom fencing scripts can be added and called here

- ***FENCE_PASSWD***

Password for fence device (only if required by the fence device)

- ***FENCE_ACTION***

Supported actions = start, stop, reset

- ***FENCE_REBOOT_LONE_HOST***

If Master host cannot see any other pool members, choose whether to reboot before attempting to fence peers. This may not matter if quorum is used (see FENCE_QUORUM_REQUIRED) 1=enabled 0=disabled

- ***FENCE_IPADDRESS***

Only used for static fencing device - currently XVM host supported. Can be used for custom fencing scripts that call a static IP such as a power strip.

- ***FENCE_HOST_FORGET***

Will be Deprecated in future release – only use if you know what you are doing. As of version 1.6.41 – this is no longer necessary. Setting this to 0 is recommended.
Select whether to forget a fenced host (permanently remove from pool) 1=enabled 0=disabled.

- ***FENCE_MIN_HOSTS***

Do not allow fencing when fewer than this number of hosts are remaining in the pool.

- ***FENCE_QUORUM_REQUIRED***

Select whether pool remaining hosts should have quorum before allowing fencing. 1=enabled 0=disabled

- ***FENCE_USE_IP_HEURISTICS***

Select whether to poll additional IP endpoints (other than the pool hosts) and possibly create an additional vote used to determine quorum for the pool. This can be helpful in a 2 node pool where quorum cannot otherwise be achieved. 1=enabled 0=disabled

- ***FENCE_HEURISTICS_IPS***

Create a list of IP addresses or domain names to check. The list should be delimited by ":"

Additional Local Configuration Parameters

Granular control over logging and email alerts is provided in the local configuration override file. These settings allow for enabling/disabling logging and email alerts on a per function basis. These can be useful when troubleshooting / logging a specific function or to minimize the volume of log files and system email alerts.

Below enables / disables logging per function 1=enable 0=disable

LOG_check_ha_enabled=1	enable / disable logging for check_ha_enabled
LOG_get_pool_host_list=1	enable / disable logging for get_pool_host_list
LOG_get_pool_ip_list=1	enable / disable logging for get_pool_ip_list
LOG_master_ip=1	enable / disable logging for master_ip
LOG_check_xapi=1	enable / disable logging for check_xapi
LOG_get_app_vms=1	enable / disable logging for get_app_vms
LOG_vm_state=1	enable / disable logging for vm_state
LOG_vm_state_check=1	enable / disable logging for vm_state_check
LOG_vm_mon=1	enable / disable logging for vm_mon
LOG_promote_slave=1	enable / disable logging for promote_slave
LOG_get_vms_on_host=1	enable / disable logging for get_vms_on_host
LOG_get_vms_on_host_local=1	enable / disable logging for get_vms_on_host_local
LOG_check_vm_managed=1	enable / disable logging for check_vm_managed
LOG_write_pool_state=0	enable / disable logging for write_pool_state ***Produces allot of logging - off by default***
LOG_check_slave_status=1	enable / disable logging for check_slave_status
LOG_fence_host=1	enable / disable logging for fence_host
LOG_email=1	enable / disable logging for email
LOG_autoselect_slave=1	enable / disable logging for autoselect_slave
LOG_check_quorum=1	enable / disable logging for check_quorum
LOG_xe_wrapper=1	enable / disable logging for xe_wrapper

Below enables / disables email alerts per function 1=enable 0=disable

MAIL_get_pool_host_list=1	enable / disable email alerts for get_pool_host_list
MAIL_get_pool_ip_list=1	enable / disable email alerts for get_pool_ip_list
MAIL_master_ip=1	enable / disable email alerts for master_ip
MAIL_check_xapi=1	enable / disable email alerts for check_xapi
MAIL_get_app_vms=1	enable / disable email alerts for get_app_vms
MAIL_vm_state=1	enable / disable email alerts for vm_state
MAIL_vm_state_check=1	enable / disable email alerts for vm_state_check

MAIL_vm_mon=1	enable / disable email alerts for vm_mon
MAIL_promote_slave=1	enable / disable email alerts for promote_slave
MAIL_get_vms_on_host=1	enable / disable email alerts for get_vms_on_host
MAIL_get_vms_on_host_local=1	enable / disable email alerts for get_vms_on_host_local
MAIL_check_vm_managed=1	enable / disable email alerts for check_vm_managed
MAIL_write_pool_state=1	enable / disable email alerts for write_pool_state
MAIL_check_slave_status=1	enable / disable email alerts for check_slave_status
MAIL_fence_host=1	enable / disable email alerts for fence_host
MAIL_email=1	enable / disable email alerts for email
MAIL_autoselect_slave=1	enable / disable email alerts for autoselect_slave
MAIL_check_quorum=1	enable / disable email alerts for check_quorum
MAIL_xe_wrapper=1	enable / disable email alerts for xe_wrapper

4. Modes of Operation

Two modes of operation are supported by HA-Lizard.

Mode 1 uses logic to manage appliances within the pool. By default, all appliances within a pool are managed by HA without requiring any additional settings. This is useful for small pools where most HA management functions can be handled from within a compatible graphical configuration utility. Simply add VMs to appliances and those VMs are automatically managed. Any VM that is not part of an appliance is not managed by HA. This technique greatly simplifies HA management as each individual VM does not require any special settings. If some appliances should not be managed by HA, simply add the UUID of the appliances to be skipped in the DISABLED_VAPPS setting.

Mode 2 provides greater flexibility by providing a mechanism for more granular control over which pool VMs are managed. Appliances are not managed in this mode. By default (when GLOBAL_VM_HA is set to 1), all VMs are automatically managed by HA. If GLOBAL_VM_HA is set to 0, then each VM within the pool must have HA explicitly enabled or disabled. This can be set from the pool GUI with access to custom configuration parameters or via the command line tool ha-cfg.

The operating mode can be configured as a global setting or override setting which allows hosts within a pool to operate in a mixed environment with some hosts managing VMs while others manage appliances. Usually the default global setting which manages VMs is adequate.

The default global settings manage all VMs within a pool and attempts to move VMs from failed hosts or start them if not running. This approach works well in most instances. If more granularity is required to select only certain VMs as managed and others ignored, a configuration setting is provided which can be set per VM. Settings can be applied using the ha-cfg command line tool or via a compatible graphical management interface with visibility into custom XAPI fields.

5. Managing HA Services

- **HA-Lizard System Service**

When installing on a CentOS based Dom0, the installer will install a startup script in /etc/init.d and set it to automatically start each time the server is started. Generally there are no additional steps required in setting up the system service.

The service can be checked, stopped or invoked with the "*service*" command which manages System V init scripts. The following arguments are supported:

"service ha-lizard start": starts the service
"service ha-lizard stop": stops the service
"service ha-lizard status": reports running status of the service

By default a watchdog service is installed and started with when installing with the included installer. Important Note: Stopping the HA service while the watchdog service is running will be ineffective as the watchdog will restart the HA service within a few seconds after a stop. The HA init script can be invoked with a "-w" option to also start or stop the watchdog service with the HA service. The –w option can be used as follows:

"service ha-lizard start –w": starts the service and watchdog
"service ha-lizard stop –w": stops the service and watchdog
"service ha-lizard status –w": reports running status of the service and watchdog

HA Watchdog Service

A watchdog service is installed by default and started/stopped via the main service init sctipt or can be individually managed with:

"service ha-lizard-watchdog start": starts the service
"service ha-lizard-watchdog stop": stops the service
"service ha-lizard-watchdog status": reports running status of the service

The default watchdog interval for checking the HA service is 10 seconds. This can be changed by editing the variable "WATCH_INTERVAL" in /etc/init.d/ha-lizard-watchdog.

Disabling the Watchdog Service

The watchdog service is mandatory when fencing is enabled. If fencing is not enabled, the watchdog service can be disabled by commenting or deleting the line "WATCHDOG=/etc/init.d/ha-lizard-watchdog" in the HA init script located in /etc/init.d/ha-lizard.

- **HA Monitor**

A separate "*monitor*" script is provided for manually starting the HA service. This is intended only for systems that are not compatible with the provided init script as described above. The default init script already includes all of the monitor logic located in this script which can be found in /etc/ha-lizard/scripts/ha-lizard.mon.

Avoid running this script if the system service is already installed as it will results in multiple monitors running that will impact performance.

The monitor is required to be running at all times for HA to work. It can be started in any of the following manners:

- Manually start the monitor and background it:
  /etc/ha-lizard/scripts/ha-lizard.mon &
- If your host runs a Dom0 operating system other than the default CentOS, an init script can be created that calls /etc/ha-lizard/scripts/ha-lizard.mon
- Add a line to /etc/rc.local which calls the monitor script on boot.
  /etc/ha-lizard/scripts/ha-lizard.mon &

- **Managing HA for the Pool**

In order for the pool to operate in HA mode, each host in the pool must be running the monitor service as described above. The monitor will periodically call the main HA-Lizard logic which executes the HA logic for the pool. A number of configurable timers and triggers can be found in the configuration settings which determine how often the monitor will invoke the HA script and when to abort. Regardless of the method chosen to run the monitor, the pool must have HA enabled to operate in HA mode.

Enabling HA is different than starting the monitor.

Two methods are provided for enabling and disabling HA for the pool:

- Enable / Disable HA via XenCenter or any compatible GUI capable of modifying a custom configuration field
- Use the CLI tool: "ha-cfg status"

Regardless of the state of HA for the pool, the monitor service will continue to run in the background on each host within the pool. The monitor will detect whether HA is enabled periodically and decide whether to provide pool level HA depending on operating status for the pool.

Cluster Management and Fencing

Cluster management and fencing with STONITH are provided with the HA-Lizard package. The behavior is configured via the global configuration settings or override settings.

- **Pool Cluster Management**

The system does not require the use of any outside cluster manager which would add more complexity to a host/pool. The overall software design tries to avoid the addition of any packages/services which would introduce complexity and potential instability into the system. Given this, built in cluster management is provided and designed to be lightweight and fairly extensible.

Cluster Failover and Recovery Logic

Node failover and recovery logic is designed around the pool concept of “Master and Slaves”. A pool can have only a single Master node and any number of Slaves. Given this, cluster management is provided in this fashion.

- Slaves cannot detect other slave failures or remove other slaves from the pool
- Only the Master host can detect a Slave Host failure and take steps to fence and remove the Slave from the pool
- Only a single Slave is allowed to detect a Master host failure. In this case the Slave can eject the former Master host and promote itself to the new pool Master. This is done intentionally to avoid a race condition where multiple Slaves try to become the Pool Master.
- The single Slave which is allowed to eject a failed Master and become the new Pool Master is dynamically chosen by the Pool Master on each invocation of HA (approximately every minute). The chosen Slave (“autopromote winner”) will have its UUID stored in the XAPI database. Slaves will then determine their status in the pool by checking this value on every invocation of HA. In the event of a Master failure, XAPI may be unresponsive, in this case Slaves will use the “autopromote winner” identity stored in local state files from the last successful iteration.
- All logic is completely dynamic and requires no static settings to determine what roles hosts play in the cluster. Each iteration of HA will detect whether a host is the Pool Master or Slave and then execute logic which specifically applies to the hosts role. Each iteration also selects a new “autopromote winner”. In the event of a Master failure and takeover by a Slave (autopromote winner), the new Master will then automatically select a new Slave as being eligible to take over as Master.

- **Split Brain Handling**

IP Heuristics can be used as an additional quorum vote when checking whether the remaining hosts in a pool have quorum to take over services and remove failed hosts. This can be especially useful in the case of a 2

node pool, where quorum would be impossible to achieve, thus preventing any HA recovery measures from being executed. Although there is no limit to the number of IP addresses that can be used, one should carefully consider the selection of useful addresses, keeping in mind that a large number of addresses will slow down any recovery procedures. Some good addresses to check would be the IP address of a shared storage pool and the IP address of a network switch which is common or shared by all the hosts in a pool. Regardless of the number of IP addresses that are checked, all of the IP addresses must by successfully reached with ICMP ping to count as a single vote. In a two node pool, this additional vote would give the surviving node enough votes to take over the services of the failed node.

- **Fencing and STONITH**

Currently, three fencing options are supported; POOL, ILO and XVM. Additionally, a structured interface is provided so that custom fencing scripts can be easily implemented.

ILO Fencing

ILO fencing is included as part of HA and developed and tested on ILO 2. The fencing logic will power off a host and return exit status verifying that the power state is off before forcefully removing a host from the pool configuration.

The following configuration parameters provide basic ILO fencing:

FENCE_ENABLED=1
FENCE_ACTION=stop
FENCE_FILE_LOC=/etc/ha-lizard/fence
FENCE_HA_ONFAIL=0
FENCE_HOST_FORGET=0
FENCE_METHOD=ILO
FENCE_MIN_HOSTS=3
FENCE_PASSWD=<ILO Password> Should be consistent across all hosts in pool or use override
FENCE_REBOOT_LONE_HOST=0
FENCE_QUORUM_REQUIRED=1

Additionally, the ILO.hosts file must be populated with UUID:IP pairs for each host in the pool. This file must not contain any spaces. Each line in the file will list <UUID of Host>:<ILO IP Address> delimited by a colon. The file should be placed in: /etc/ha-lizard/fence/ILO and named ILO.hosts. Capitalization and spelling are important as the file names are dynamically built using the HA configuration parameters.

To edit/create the file, type the following from the command line:

vi /etc/ha-lizard/fence/ILO/ILO.hosts
Alternatively, your preferred text editor can be used instead of “vi”

Sample file is presented below:

```
2f4a4020-fe8c-4686-a65d-501a19a06716:192.168.1.141
33e68e2c-5b7b-4d5e-ae9c-7baef5b1566b:192.168.1.132
```

Pool Fencing

Pool fencing is “dummy” fence device that does not provide STONITH (as in ILO). It is intended to provide Pool level fencing which will forcefully remove any unresponsive hosts from the pool but will not attempt to power off the host.

The following configuration parameters provide basic Pool fencing:

FENCE_ACTION=stop
FENCE_ENABLED=1
FENCE_FILE_LOC=/etc/ha-lizard/fence
FENCE_HA_ONFAIL=1
FENCE_HOST_FORGET=0
FENCE_IPADDRESS= (IP is not necessary for POOL fence method, leave blank)
FENCE_METHOD=POOL
FENCE_MIN_HOSTS=3
FENCE_PASSWD= (Password is not necessary for POOL fence method, leave blank)
FENCE_REBOOT_LONE_HOST=0
FENCE_QUORUM_REQUIRED=1

XVM Fencing

XVM fencing is intended for test environments and development environments where a large pool is simulated within a single physical host. XVM fencing will fence pool hosts that are virtual machines within a single physical host, meaning, a single DomO will host a number of DomUs, each of which is running XCP or XenServer and are part of a pool.

The following configuration parameters provide basic XVM fencing:

FENCE_ACTION=stop
FENCE_ENABLED=1
FENCE_FILE_LOC=/etc/ha-lizard/fence
FENCE_HA_ONFAIL=1
FENCE_HOST_FORGET=0
FENCE_IPADDRESS=<IP Address of DomO>
FENCE_METHOD=XVM
FENCE_MIN_HOSTS=3
FENCE_PASSWD=<root SSH password of DomO>
FENCE_REBOOT_LONE_HOST=0
FENCE_QUORUM_REQUIRED=1

Additionally, the XVM.hosts file must be populated with UUID:UUID pairs for each host in the pool. This file must not contain any spaces. Each line in the file will list <UUID of Pool HOST>:<UUID of VM in DomO> delimited by a colon. The file should be placed in: /etc/ha-lizard/fence/XVM and named XVM.hosts. Capitalization and spelling are important as the file names are dynamically built using the HA configuration parameters.

To edit/create the file, type the following from the command line:

vi /etc/ha-lizard/fence/XVM/XVM.hosts
Alternatively, your preferred text editor can be used instead of “vi”

Sample file is presented below:

```
9b9a243a-abf3-485f-bb37-e43b29555cbe:0838afcc-1c6c-549b-a28c-d7916ca7753a
1ae14847-6169-4cb0-ad8b-79ef0219e1ee:2372bc81-0709-c8e8-6e34-3ccb085cf1ab
8b8470a5-8848-41a4-bf31-7ce6f1c7ea5e:2dcea2bd-b47c-cee3-fcda-c7b6f3953555
c8ee6f60-71fe-442f-9002-92b16f35f47e:33d4f85a-72d8-2d7a-64b0-26c77dd4b916
```

- **Adding your own Custom Fencing Script**

A structured framework is provided for adding custom fencing scripts. The general parameters for adding a custom script are as follows:

- Choose a name for the custom fencing script. This name will be placed into the pool HA configuration and used to dynamically build file names and references. For example, for a custom fencing agent named “POWEROFF” set the fencing method configuration parameter as follows:

  FENCE_METHOD=POWEROFF

  this can be set with the command line configuration tool: “ha-cfg set FENCE_METHOD POWEROFF”

- Create a folder with the same *exact* name in /etc/ha-lizard/fence/

  mkdir /etc/ha-lizard/fence/POWEROFF

- Place your custom fencing agent in the newly created folder. The agent MUST be named POWEROFF.sh. If your agent is not a bash script and named something else like POWEROFF.py, then simply create a wrapper named POWEROFF.sh. Your script must accept the following passed in arguments:

  arg1 = UUID of the host to be fenced. Your script should employ some logic to convert this UUID to some useful data like the IP address of the host or the location of a power outlet for the given host UUID

  arg2 = Fence Action. Supported actions (configured in the global configuration) are:
  start, stop, reset

- Your script must return an exit status after executing.
  return “0” on success for the passed in action
  return “1” on general failure for the passed in action

6. Managing via XenCenter

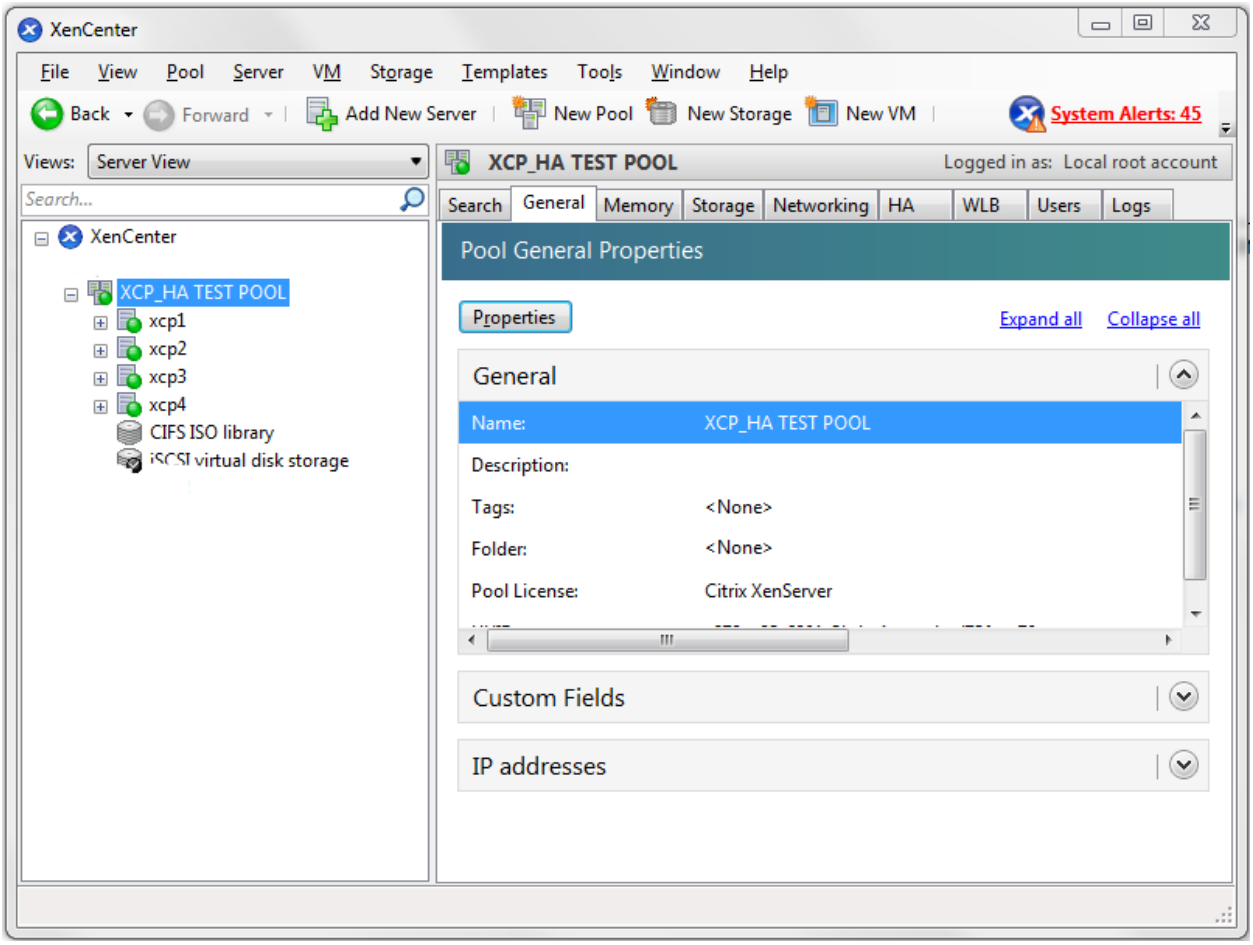
Once installed, the HA service can be toggled on / off via the command line tool, ha-cfg, or via Citrix's XenCenter for users that prefer to use a graphical tool.

XenCenter custom fields can be added for:

- Enabling / Disabling HA for the entire pool
- Enabling / Disabling HA for specific VMs within the pool (used when GLOBAL_VM_HA is set to 0)

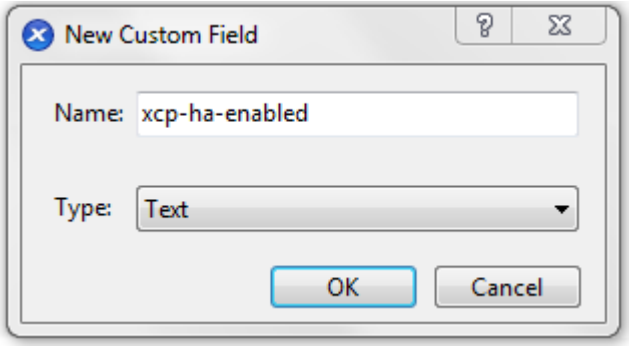
The XenCenter friendly custom fields were introduced into the pool database during installation of HA-Lizard. Follow these steps to make these fields visible from within XenCenter:

1) Highlight the pool name in the left navigation window of XenCenter and select the General tab



2) Select Properties -> Custom Fields -> Edit Custom Fields -> Add
Add a new custom field with the ***exact*** name as the pool configuration setting XC_FIELD_NAME. By

default this name is “ha-lizard-enabled” and create a new field with type set to “text”.



3) Any settings that were previously set via the command line tool should now be visible within the new XenCenter custom field. The custom field will be automatically present at both the Pool level and individual VMs and can be used in the following manner:

- To Enable or Disable HA for the entire pool. Modify the value of the custom field at the Pool level as follows:

  set to “true” to enable HA for the Pool
  set to “false” to disable HA for the Pool

  Be aware that the entry is not validated when entered via XenCenter. The CLI tool can be used to validate the setting with “ha-cfg status”. Additionally, changing the status via the CLI will also change the status viewed within XenCenter. The CLI tool does validate entries to avoid mis-spelling “true” or “false” and is the preferred method to enable/disable HA.

- To Enable or Disable HA for a specific VM. Modify the value of the custom field for the VM as follows:

  set to “true” to enable HA for the VM
  set to “false” to disable HA for the VM

  Be aware that the entry is not validated when entered via XenCenter. The CLI tool can be used to validate the setting with “ha-cfg get-vm-ha”. Additionally, changing the status via the CLI will also change the status viewed within XenCenter. The CLI tool does validate entries to avoid mis-spelling “true” or “false” and is the preferred method to enable/disable HA for VMs. Use the CLI command “ha-cfg set-vm-ha” to enbale/disable HA from the command line.

4) **Deleteting the custom field from XenCenter will delete the HA settings for the pool. *Do not delete the custom field once created.***

7. Application and Settings Examples

- **Auto-Start VMs – Without Host High Availability**

The following configuration settings can be used for a basic setup where VMs are automatically started in either of the following scenarios:

1) A VM has failed during normal operation

2) A host was rebooted and VMs on the host require a restart

In this mode, HA is disabled and any host failures are ignored. In the event of a host failure, some VMs may fail to start properly while the host is still part of the pool.

```
DISABLED_VAPPS=()
ENABLE_LOGGING=1
FENCE_ACTION=stop
FENCE_ENABLED=0
FENCE_FILE_LOC=/etc/ha-lizard/fence
FENCE_HA_ONFAIL=0
FENCE_HEURISTICS_IPS=
FENCE_HOST_FORGET=0
FENCE_IPADDRESS=
FENCE_METHOD=POOL
FENCE_MIN_HOSTS=3
FENCE_PASSWD=
FENCE_QUORUM_REQUIRED=0
FENCE_REBOOT_LONE_HOST=0
FENCE_USE_IP_HEURISTICS=0
GLOBAL_VM_HA=1
MAIL_FROM=<enter email address>
MAIL_ON=1
MAIL_SUBJECT="SYSTEM_ALERT-FROM_HOST:$HOSTNAME"
MAIL_TO=<enter email address>
MONITOR_DELAY=45
MONITOR_KILLALL=1
MONITOR_MAX_STARTS=40
MONITOR_SCANRATE=10
OP_MODE=2
PROMOTE_SLAVE=0
SLAVE_HA=0
SLAVE_VM_STAT=1
XAPI_COUNT=5
XAPI_DELAY=15
XC_FIELD_NAME='ha-lizard-enabled'
XE_TIMEOUT=10
```

- **Auto-Start VMs – With Host High Availability**

The following configuration settings can be used for a basic HA setup where VMs are automatically started in any of the following scenarios:

1) A VM has failed during normal operation
2) A host was rebooted and VMs on the host require a restart
3) A host (master or slave) has failed and had VMs running before the failure

Additionally, any host failure will be detected by HA with steps taken to automatically recover and reconfigure the pool, starting any affected services.

In this mode, HA is enabled with POOL fencing, meaning, any failed host will be detected and forcibly removed from the pool. Additional configuration possibilities are available using ILO or custom fencing agents.

```
DISABLED_VAPPS=()
ENABLE_LOGGING=1
FENCE_ACTION=stop
FENCE_ENABLED=1
FENCE_FILE_LOC=/etc/ha-lizard/fence
FENCE_HA_ONFAIL=1
FENCE_HEURISTICS_IPS=<enter IP addresses delimited by :>
FENCE_HOST_FORGET=0
FENCE_IPADDRESS=
FENCE_METHOD=POOL
FENCE_MIN_HOSTS=3
FENCE_PASSWD=
FENCE_QUORUM_REQUIRED=1
FENCE_REBOOT_LONE_HOST=1
FENCE_USE_IP_HEURISTICS=1
GLOBAL_VM_HA=1
MAIL_FROM=<enter email address>
MAIL_ON=1
MAIL_SUBJECT="SYSTEM_ALERT-FROM_HOST:$HOSTNAME"
MAIL_TO=<enter email address>
MONITOR_DELAY=45
MONITOR_KILLALL=1
MONITOR_MAX_STARTS=40
MONITOR_SCANRATE=10
OP_MODE=2
PROMOTE_SLAVE=1
SLAVE_HA=1
SLAVE_VM_STAT=1
XAPI_COUNT=5
XAPI_DELAY=15
XC_FIELD_NAME='ha-lizard-enabled'
XE_TIMEOUT=10
```

- **2-Node Pool – With Host High Availability**

A 2-node pool is a special case where IP heuristics are required to create an additional vote to achieve quorum. This technique tries to reach IP addresses outside of the pool should a peer host be unreachable. The following settings provide full HA features in a 2-node environment. For faster switching of roles and HA recovery consider shortening XAPI_COUNT and XAPI_DELAY.

```
DISABLED_VAPPS=()
ENABLE_LOGGING=1
FENCE_ACTION=stop
FENCE_ENABLED=1
FENCE_FILE_LOC=/etc/ha-lizard/fence
FENCE_HA_ONFAIL=1
FENCE_HEURISTICS_IPS=<enter IP addresses delimited by :>
FENCE_HOST_FORGET=0
FENCE_IPADDRESS=
FENCE_METHOD=POOL
FENCE_MIN_HOSTS=2
FENCE_PASSWD=
FENCE_QUORUM_REQUIRED=1
FENCE_REBOOT_LONE_HOST=0
FENCE_USE_IP_HEURISTICS=1
GLOBAL_VM_HA=1
MAIL_FROM=<enter email address>
MAIL_ON=1
MAIL_SUBJECT="SYSTEM_ALERT-FROM_HOST:$HOSTNAME"
MAIL_TO=<enter email address>
MONITOR_DELAY=45
MONITOR_KILLALL=1
MONITOR_MAX_STARTS=40
MONITOR_SCANRATE=10
OP_MODE=2
PROMOTE_SLAVE=1
SLAVE_HA=1
SLAVE_VM_STAT=1
XAPI_COUNT=5
XAPI_DELAY=15
XC_FIELD_NAME='ha-lizard-enabled'
XE_TIMEOUT=10
```

8. Miscellaneous

- **Dependencies and Compatibility**

When installing HA onto a default Centos based DomO (XCP or XenServer), all the required tools needed to run HA are resident on the system with the exception of sendmail and mailx, which, are not mandatory and only needed if email alerts are required.

Package is compatible with XCP version 1.6 and XenServer version 6.1. Prior releases may work but have not been tested.

For custom DomO installations, ensure the following tools are available:

xapi and xe toolstack
/bin/awk
/bin/echo
/bin/logger
/bin/hostname
/bin/mail (only if email alerts are required)
/opt/xensource/sm/resetvdis.py
/bin/cat
/bin/grep
/usr/bin/column
/etc/init.d/functions

- **Security and Ports**

- HTTP port 80 is used to check the running status of XAPI on hosts within the pool. Ensure that all hosts within the pool have access to port 80 on all peer hosts.

- ICMP (ping) is used to check the running status of peer hosts. Ensure that all hosts within the pool have access to ICMP on all peer hosts.

- **Things to Know**

- ***FENCE_HOST_FORGET will be deprecated. Avoid setting this to 1 unless you know what you are doing. This will not impact recovery of a pool, but, enabling will forcefully remove a host from the pool. The host would then have to be manually repaired and reintroduced into the pool. Version 1.41 introduced new friendlier logic which no longer requires the forceful removal of a host.***

- Hosts should be put into maintenance mode to prevent them from being fenced and removed from the pool while being worked on. Any host in maintenance mode will be safely skipped over by the HA logic.

- HA should be disabled for a pool before shutting down a host. Do not shut down a host without disabling HA or host can be forcefully removed from the pool.

- Hostname is used to determine the local UUID of a host. Server hostname must match the XAPI name-label for each host. This is the default scenario for XCP/XenServer – avoid changing server host names or portions of HA will fail.

- Fenced Hosts are removed ("forgotten") from the pool. This can be destructive if hosts are using local storage. The HA environment should be comprised of hosts utilizing shared backend storage (ex. iSCSI) only. Hosts with VMs running on local storage should not be used in HA environments. A script for cleaning up forcefully removed hosts can be found in /etc/ha-lizard/scripts/.
- Disabling ha-lizard: This can be accomplished with a custom-field accessible via a graphical management utility or the local CLI tool, ha-cfg. IMPORTANT: slaves read ha-lizard-enabled from local state files and may take up to a minute before the cache is updated.

- **Future Improvements**

- Significant performance boost with centralized state data collection and distribution vs current model which utilizes a distributed method with redundant calls.
- Improved fencing support.
- Parallelize fencing calls so that multiple hosts can be fenced simultaneously. Currently this is a serial operation, however, it is a rare condition to have multiple hosts fail simultaneously in a properly designed network.
- Add VM start priority
- Email alert banning to avoid floods of repeated email alerts. Logic already introduced in iSCSI-HA addon will be moved to the HA-Lizard project.

- **Support**

- Post a question on the support forum
  http://www.halizard.com/index.php/forum
- Contact the project sponsor
  http://www.pulsesupply.com